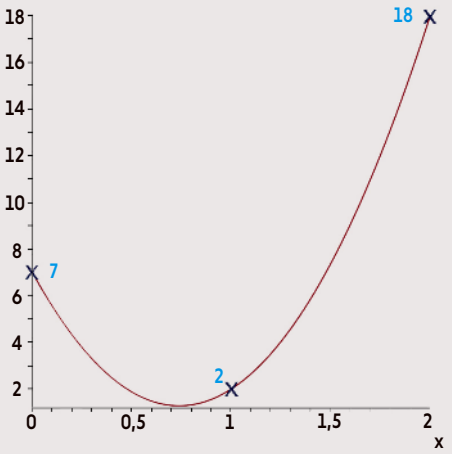
2 L'INTERPOLATION



Imaginons que l'on veuille trouver un polynôme P(x) qui prenne la valeur 7 pour $x = 0$, la valeur 2 pour $x = 1$, et la valeur 18 pour $x = 2$. Voici une méthode bien connue qui permet de le construire et qui se généralise à un nombre quelconque de données.

On écrit : $P(x) = 7(x-1)(x-2)/[(0-1)(0-2)] + 2(x-0)(x-2)/[(1-0)(1-2)] + 18(x-0)(x-1)/[(2-0)(2-1)]$.

Le premier terme est nul pour $x = 1$ et $x = 2$, le second pour $x = 0$ et $x = 2$, le troisième pour $x = 0$ et $x = 1$. Donc, seul le premier terme n'est pas nul pour $x = 0$, seul le second n'est pas nul pour $x = 1$, seul le troisième n'est pas nul pour $x = 2$.

De plus, les facteurs 7, 2 ou 18 au début de chaque terme assurent que le terme non nul pour $x = 0$ prenne la valeur 7, pour $x = 1$ prenne la valeur 2, pour $x = 2$ prenne la valeur 18. C'est ce que l'on recherchait. En développant et en simplifiant, on trouve :

$P(x) = 21x^2/2 - 31x/2 + 7$.

On vérifie aisément que $P(0) = 7$, $P(1) = 2$, $P(2) = 18$. La méthode se généralise et permet, si $y_0, y_1, \ldots, y_n$ sont des valeurs données, de construire sans mal un polynôme qui prendra la valeur y_0 pour $x = 0$, y_1 pour $x = 1$, ..., y_n pour $x = n$. Telle est la « méthode d'interpolation de Lagrange ».

Elle montre qu'autoriser beaucoup de paramètres, ici les coefficients du polynôme, permet sans mal de faire passer un polynôme par les valeurs souhaitées, quelles qu'elles soient.

Cependant, en général, la fonction trouvée oscillera beaucoup, alors qu'on préférerait une fonction aussi régulière et stable que possible : son ajustement aux valeurs voulues est exact, mais insatisfaisant. Il existe de meilleures méthodes permettant d'approcher les valeurs voulues en limitant les oscillations de la fonction résultat.

Retenons qu'interpoler exactement est facile, mais que cela ne satisfait pas en général le désir de trouver des lois lisses et régulières.

Cette logique de l'interpolation de Lagrange est poussée jusqu'à l'absurde dans les résultats présentés dans le texte principal : la méthode découverte par Steven Piantadosi, bien plus puissante que l'interpolation de Lagrange, permet, en changeant un seul paramètre p d'une fonction $f_p(x)$, d'obtenir que pour des valeurs quelconques $y_0, y_1, \ldots, y_n$, on ait :

$f_p(0) = y_0$, $f_p(1) = y_1$, ..., $f_p(n) = y_n$.

> fonction $f_p(x)$ pour $x = 0, 1, \ldots, n$ les fera remonter une à une : quand x vaudra 0, y_0 remontera, quand x vaudra 1, y_1 remontera, etc.

L'encadré 3 explique que cette remontée des valeurs approchées de $y_0, y_1, \ldots, y_n$ s'opère indirectement grâce à la fonction $2x$. Quand on applique la fonction $2x$ à un nombre réel écrit en base 2, par exemple $x = 0{,}110011101111_2$, elle fait remonter tous les chiffres vers la gauche : $2x = 1{,}10011101111_2$ (le 2 en indice indique qu'il s'agit d'une écriture en base 2).

Cette propriété est équivalente, pour la base 2, à ce que l'on connaît bien en base 10 : multiplier par 10 décale toutes les décimales d'un chiffre vers la gauche. Par exemple, $10 \times 3{,}1415926\ldots = 31{,}415926\ldots$

En utilisant la fonction qui enlève ce qui est devant la virgule, la fonction (x mod 1), parfois appelée « partie fractionnaire de x », on a alors un moyen de faire remonter les chiffres binaires d'un nombre x donné en effaçant ceux qui sont placés en tête. Posons en effet $D(x) = 2x$ mod 1, et prenons par exemple $x = 0{,}11001110111101012$. On a alors :

$D(x) = 0{,}10011101111010_2$,

$D(D(x)) = 0{,}00111011110101_2$,

$D(D(D(x))) = 0{,}01110111101012$, etc.

En exploitant ce moyen de cacher des informations dans un paramètre réel p, ainsi que quelques autres astuces (*voir l'encadré 4*), on arrive à la fonction qui, avec le bon choix de p, donne y_0 pour $x = 0$, y_1 pour $x = 1$, ..., y_n pour $x = n$ avec une précision fixée à l'avance. Cela montre qu'un seul paramètre p et une seule fonction $f_p(x)$ permettent d'approcher toutes les données que l'on veut, quel que soit leur nombre, pourvu qu'il soit fini !

Ainsi, pas besoin de quatre paramètres pour dessiner un éléphant : avec un paramètre unique, on peut s'arranger, et la même fonction servira toujours, en changeant p, pour dessiner un poisson, une tortue, un oiseau ou tout ce que l'on veut.

NOMBRE DE PARAMÈTRES ET COMPLEXITÉ DES MODÈLES

On mesure parfois la complexité d'un modèle en comptant le nombre de paramètres utilisés par le modèle. C'est une façon de procéder qui ne peut pas se justifier sans précisions supplémentaires sur la nature du modèle. En effet, la fonction $f_p(x)$ de Steven Piantadosi montre que l'on peut toujours se ramener à un seul paramètre. Si l'on veut donner un sens au décompte des paramètres d'un modèle, il faut détailler les contraintes qu'on impose au modèle, par exemple « ne faire intervenir que des polynômes » ce qui interdira d'accepter la fonction $f_p(x)$ ou l'une de ses variantes.

Il faut insister sur l'idée que l'article de Steven Piantadosi ne prétend nullement

proposer une méthode nouvelle pour construire des modèles approchant des données en utilisant une fonction à un seul paramètre (*voir l'encadré 5*). L'article est seulement un jeu mathématique qui exploite la capacité qu'a un nombre réel de détenir une quantité infinie d'information. Ce qui fait l'intérêt particulier de l'article et le rend étonnant est la simplicité globale de la méthode. Je ne crois pas que l'on ait publié antérieurement au travail de Steven Piantadosi une méthode aussi élémentaire permettant de déterminer et calculer effectivement un paramètre p qui accumule des informations en nombre illimité, et de faire remonter ce qu'on a caché dans p en n'utilisant qu'une seule fonction construite à partir des fonctions élémentaires de l'analyse classique (sin, arcsin, exponentielle, etc.).

Les physiciens utilisent des nombres réels, mais sans jamais prendre en compte plus de 20 ou 30 décimales (*voir l'article de cette rubrique de décembre 2019*). Aussi ne donneront-ils aucun sens à des modèles comme celui suggéré par la fonction prodigieuse de Steven Piantadosi, qui ne fonctionne que si l'on accepte de manipuler des centaines de chiffres significatifs.

Si l'on examine de près la fonction $f_p(x)$ quand le paramètre p a été fixé, on s'aperçoit qu'elle oscille violemment et que les valeurs qu'elle prend aux nombres entiers 0, 1, ..., n ne sont en rien représentatives de ce qui se passe pour les x compris entre 0 et 1, entre 1 et 2, entre 2 et 3, etc.

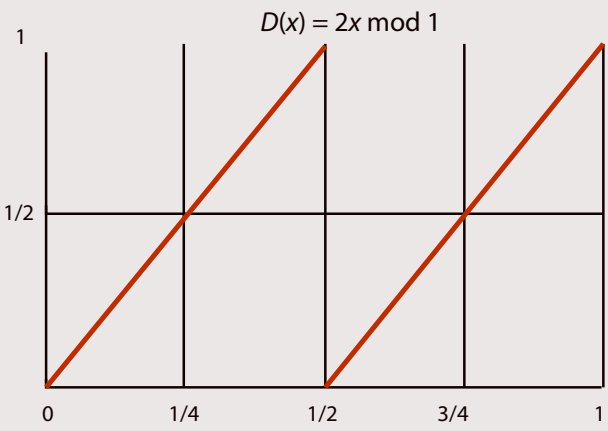
Si l'on dessine (autant qu'on peut le faire) tout le tracé de la fonction $f_p(x)$ entre 0 et n, on ne voit pas du tout l'image que produit le dessin quand on ne retient que les valeurs de $f_p(x)$ pour $x = 0, 1, ..., n$. On voit un zigzag serré qui noircit presque tout le rectangle $[0, n] \times [0, 1]$. Le tracé de la fonction $f_p(x)$ passe bien tout près des points de coordonnées $(0, y_0)$, $(1, y_1)$, ..., (n, y_n), mais il le fait sans limiter ses variations, alors que c'est ce qu'on attend d'une bonne fonction d'interpolation: on désire cette dernière aussi lisse que possible.

Citons les commentaires de Steven Piantadosi concernant sa fonction miraculeuse:

L'existence d'une fonction aussi simple avec une telle liberté de comportement illustre un problème plus fondamental. Elle montre que la complexité d'un modèle ne peut pas être évaluée en comptant ses paramètres.

Plus généralement, l'utilisation non critique d'une approche par dénombrement de paramètres ignore qu'un seul paramètre à valeur réelle contient potentiellement une quantité illimitée d'informations puisque, pour spécifier entièrement un nombre réel, un nombre infini de bits doit l'être, ce qui d'ailleurs est en soi problématique. Un autre fait notable est que l'ensemble des nombres réels >

3 UNE PREMIÈRE FONCTION MIRACULEUSE



Soit D la fonction telle que $D(x) = 2x$ pour $0 \leq x < 1/2$ et $D(x) = 2x - 1$ pour $1/2 \leq x < 1$. Cela peut écrire : $D(x) = 2x \bmod 1$.

La fonction x mod 1, parfois appelée partie fractionnaire de x, ne garde que ce qui est après la virgule : 2,37 devient 0,37 ; 219,2222 devient 0,2222.

Appliquée à un nombre écrit en binaire, par exemple $x = 0{,}11011110_2$, la fonction $D(x)$ donne un nombre ayant le même développement binaire que x, sauf que le premier chiffre après la virgule a disparu : $D(0{,}11011110_2) = 0{,}1011110_2$. L'explication est simple. D'une part, $2x$ décale le développement binaire d'un chiffre vers la gauche (c'est comme en base 10 quand on multiplie un nombre par 10), et donc $0{,}11011110_2$ devient $1{,}1011110_2$; d'autre part, $2x$ mod 1 enlève le 1 devant la virgule s'il y en a un.

Considérons maintenant la fonction D^k obtenue en appliquant k fois de suite la fonction D : $D^k(x) = D(D(...(D(x)...))$, k fois. La fonction $D^k(x)$ fait glisser les chiffres binaires de x de k places vers la gauche et élimine les k premiers chiffres : $D^k(x) = 2^k x \bmod 1$. Par exemple, $D^4(0{,}11011110_2) = 0{,}1110_2$.

Soient des nombres $y_0, y_1, ..., y_n$ compris entre 0 et 1, chacun écrit en base 2 avec r chiffres après la virgule. Exemple pour $r = 5$ et $n = 2$:
$y_0 = 0{,}10111_2$; $y_1 = 0{,}11001_2$;
$y_2 = 0{,}01011_2$
Soit $p = 0{,}1011111001010111_2$ obtenu en mettant les uns derrière les autres les chiffres de y_0, y_1 et y_2.
On a :
$p = D^0(p) = 0{,}1011111001010111_2$
$D^1(p) = 0{,}0111110010101_2$
$D^2(p) = 0{,}1111100101011_2$
$D^3(p) = 0{,}1111001010111_2$
$D^4(p) = 0{,}1110010101_2$
$D^5(p) = 0{,}1100101011_2$, etc.
Ainsi, $D^0(p)$ correspond à y_0 sur les 5 premiers chiffres binaires, $D^5(p)$ correspond à y_1 sur les 5 premiers chiffres binaires, $D^{10}(p)$ correspond à y_2 sur les 5 premiers chiffres binaires, etc.

Ce procédé place donc dans les chiffres binaires de p les chiffres binaires des données y_0, y_1,..., y_n avec une précision de r chiffres binaires pour chacun, ce qui permet, en appliquant r fois de suite la fonction D, d'avoir successivement y_0, y_1,..., y_n, à chaque fois avec une précision de r chiffres binaires.

En résumé, p contient, cachés dans ses chiffres binaires, les nombres $y_0, y_1, ..., y_n$ avec une précision de r chiffres binaires pour chacun, et la fonction D, appliquée de manière répétée à p, retrouve ces nombres cachés.

Soit maintenant la fonction E_p définie par $E_p(x) = D^{rx}(p)$, dont la variable est x et qui dépend du paramètre p. On a :
$E_p(0) = y_0$ avec une précision de r chiffres binaires,
$E_p(1) = y_1$ avec une précision de r chiffres binaires,...,
$E_p(n) = y_n$ avec une précision de r chiffres binaires.

En choisissant bien le paramètre p, la fonction $E_p(x)$ est donc capable de redonner les nombres $y_0, y_1,..., y_n$ avec une précision de r chiffres binaires. Si les données ne sont pas comprises entre 0 et 1, on s'y ramène en divisant tout par un nombre assez grand.

Ainsi, en ajustant bien son paramètre p, la fonction $E_p(x)$ suffit pour retrouver tout jeu fini de données. Ce n'est pas le résultat annoncé concernant la fonction $f_p(x) = \sin^2(2^{kx} \arcsin(\sqrt{p}))$, mais nous y sommes presque. L'encadré 4 explique comment y parvenir.